Package: jsonedit (via r-universe)

January 9, 2026

Type Package

Title JSON Parsing and Modification Utilities

Version 1.0.0

Description Bindings to 'node-jsonc-parser' to parse, format, and modify JSON files while retaining comments.

License MIT + file LICENSE

URL https://github.com/r-lib/jsonedit

Imports cli, rlang (>= 1.1.0), V8 (>= 6.0.6)

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Config/pak/sysreqs libssl-dev libnode-dev

Repository https://test.r-universe.dev

Date/Publication 2026-01-09 17:19:29 UTC

RemoteUrl https://github.com/r-lib/jsonedit

RemoteRef HEAD

RemoteSha 0b169c2a3f5a3609296c4eea47519a23599691ab

Contents

format *Format a JSON file or string*

Description

Format a JSON file or string, preserving comments.

Usage

```
text_format(text, ..., formatting_options = NULL)

file_format(file, ..., formatting_options = NULL)

formatting_options(
  indent_width = 4L,
  indent_style = "space",
  eol = "\n",
  insert_final_newline = TRUE
)
```

Arguments

text	A single string containing JSON.
...	These dots are for future extensions and must be empty.
formatting_options	The result of formatting_options(). If NULL, a default set of options are used.
file	Path to file on disk. File must exist.
indent_width	The number of spaces to use to indicate a single indent when indent_style = "space".
indent_style	The style of indentation to use. Either: • "space" for spaces. • "tab" for tabs.
eol	The character used for the end of a line. This is only applicable when the text doesn't already contain an existing line ending, i.e. an empty string or a string spanning a single line.
insert_final_newline	Whether or not to insert a final newline.

Examples

```
text <- '{"foo":[1,2]}'
cat(text_format(text))

formatting_options <- formatting_options(indent_width = 2)
cat(text_format(text, formatting_options = formatting_options))
```

`modify` *Modify a JSON file or string*

Description

Set or delete fields in a JSON file or string while retaining comments and whitespace.

Usage

```
text_modify(
  text,
  path,
  value,
  ...,
  parse_options = NULL,
  modification_options = NULL
)

file_modify(
  file,
  path,
  value,
  ...,
  parse_options = NULL,
  modification_options = NULL
)

modification_options(formatting_options = NULL, is_array_insertion = FALSE)
```

Arguments

`text`	A single string containing JSON to modify.
`path`	Either: • A character vector representing a path to a JSON element by name, i.e. `c("[r]", "editor.formatOnSave")`. • A list of strings or numbers representing a path to a JSON element by name and position, i.e. `list("[r]", "editor.rulers", 2)`. Numeric positions are specified as positive integers and are only applicable for arrays. `-1` is specially recognized as a request to *insert* at the end of an array.
`value`	New value. Wrap in `V8::JS()` to specify literal JavaScript value. Use `NULL` to delete the field.
`...`	These dots are for future extensions and must be empty.
`parse_options`	The result of `parse_options()`. If `NULL`, a default set of options are used.

modification_options
: The result of `modification_options()`. If `NULL`, a default set of options are used.

file
: Path to file on disk. File must exist.

formatting_options
: The result of a call to `formatting_options()`. If `NULL`, a default set of options are used.

is_array_insertion
: Whether or not to treat the change as an *insertion* at the specified `path` rather than a *modification* at that `path`. Only applicable for arrays.

Examples

```
text <- "{}"

text <- text_modify(text, c('[r]', 'editor.formatOnSave'), TRUE)
cat(text)

text <- text_modify(text, c('[r]', 'editor.formatOnSave'), NULL)
cat(text)

# Insert an array
text <- text_modify(text, "foo", 1:3)
cat(text)

# Update the array at location 2
cat(text_modify(text, list("foo", 2), 0))

# Insert at location 2
cat(text_modify(
  text,
  list("foo", 2),
  0,
  modification_options = modification_options(is_array_insertion = TRUE)
))

# Insert at the end of the array. `-1` is treated as an insertion regardless
# of the value of `is_array_insertion`.
cat(text_modify(text, list("foo", -1), 0))

# Only the modified elements are reformatted
text <- '{"foo":[1,2],\n"bar":1}'
cat(text_modify(text, list("foo", 3), 0))

# You can control how those elements are formatted
cat(text_modify(
  text,
  list("foo", 3),
  0,
  modification_options = modification_options(
    formatting_options = formatting_options(indent_width = 2),
    is_array_insertion = TRUE
```

```
  )
))
```

`parse` *Parse a JSON file or string*

Description

- `text_parse()` and `file_parse()` parse JSON into an R object.
- `text_parse_at_path()` and `file_parse_at_path()` parse JSON at a requested JSON path, i.e. `c("[r]", "editor.formatOnSave")`.

Usage

```
text_parse(text, ..., parse_options = NULL)

file_parse(file, ..., parse_options = NULL)

text_parse_at_path(text, path, ..., parse_options = NULL)

file_parse_at_path(file, path, ..., parse_options = NULL)

parse_options(
  allow_comments = TRUE,
  allow_trailing_comma = TRUE,
  allow_empty_content = TRUE
)
```

Arguments

`text`	A single string containing JSON.
`...`	These dots are for future extensions and must be empty.
`parse_options`	The result of `parse_options()`. If `NULL`, a default set of options are used.
`file`	Path to file on disk. File must exist.
`path`	Either: • A character vector representing a path to a JSON element by name, i.e. `c("[r]", "editor.formatOnSave")`. • A list of strings or numbers representing a path to a JSON element by name and position, i.e. `list("[r]", "editor.rulers", 2)`. Numeric positions are specified as positive integers and are only applicable for arrays.
`allow_comments`	Whether or not to allow comments when parsing.
`allow_trailing_comma`	Whether or not to allow a trailing comma when parsing.
`allow_empty_content`	Whether or not to allow empty strings or empty files when parsing.

Examples

```
text <- '
{
  "a": 1,
  "b": [2, 3, 4],
  "[r]": {
    "this": "setting",
    // A comment!
    "that": true
  }, // A trailing comma!
}
'

# Parse the JSON, allowing comments (i.e. JSONC)
str(text_parse(text))

# Try to parse the JSON, but comments aren't allowed!
parse_options <- parse_options(allow_comments = FALSE)
try(text_parse(text, parse_options = parse_options))

# Try to parse the JSON, but trailing commas aren't allowed!
parse_options <- parse_options(allow_trailing_comma = FALSE)
try(text_parse(text, parse_options = parse_options))

# Parse only a subset of the JSON
text_parse_at_path(text, "b")
text_parse_at_path(text, "[r]")
text_parse_at_path(text, c("[r]", "that"))

# Use a `list()` combining strings and positional indices when
# arrays are involved
text_parse_at_path(text, list("b", 2))
```

Index